BCDataWorks

GRADİENT BOOSTİNG KÜTÜPHANELERİ

XGBoost
LightGBM
CatBoost

Üç kütüphaneyi gerçekten ayıran tasarım kararları, parametre karşılıkları ve kredi riski analitiğinde hangisinin ne zaman kazandığı.

· Tasarım farkları · Parametre eşlemesi · Kredi riski kıyası

Bu doküman ne anlatıyor?

Üçü de aynı işi yapar: ardışık karar ağaçlarıyla gradient boosting. Fark, aynı problemi çözerken verdikleri **üç tasarım kararında** saklıdır. Bu doküman o kararları görselleştirir, parametreleri karşılıklı eşler ve kredi riski verisinde hangisinin neden öne çıktığını gösterir.

Üç cümlede fark

- **XGBoost** — dengeli ağaçlar büyütür; olgun, öngörülebilir ve kısıt desteği en geniş olan referans noktasıdır.
- **LightGBM** — en çok kazanç veren yaprağı büyütür; büyük veride en hızlısıdır, karşılığında aşırı öğrenmeye en yatkınıdır.
- **CatBoost** — her seviyede aynı bölünmeyi kullanan simetrik ağaçlar kurar; kategorik değişkenleri sızıntısız işler, varsayılanları en güvenlidir.

İçindekiler

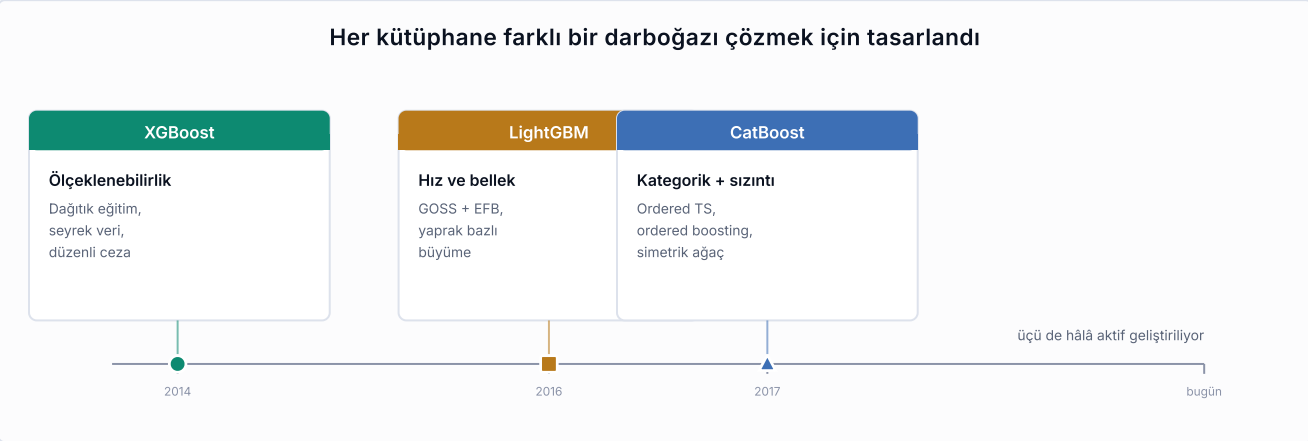
Sürüm ve ölçüm notu

Anlatım **XGBoost 3.x**, **LightGBM 4.x** ve **CatBoost 1.2.x** sürüm aileleri esas alınarak yazılmıştır. Varsayılan değerler sürümler arasında değişebilir; kritik bir karar vermeden önce kullandığın sürümün dokümantasyonundan doğrula.

Şekillerdeki süre, bellek ve performans sayıları **temsilîdir** — tipik davranışın yönünü göstermek içindir, benchmark sonucu değildir. Kendi verinde ölçmeden karar verme; 16. bölüm bunun protokolünü verir.

01 Üç kütüphane tek bakışta

Üçü de gradient boosted decision tree ailesindendir. Farkları, ortaya çıktıkları dönemde çözmeye çalıştıkları problemden gelir.



Şekil 1. Ortaya çıkış sırası ve her birinin öncelikli tasarım hedefi.

	XGBoost	LightGBM	CatBoost
Geliştirici / yıl	DMLC, 2014	Microsoft, 2016	Yandex, 2017
Ağaç büyütme	Seviye bazlı (depthwise); lossguide seçeneği var	Yaprak bazlı (best-first), num_leaves ile sınırlı	Simetrik / oblivious — seviyede tek bölünme
Kapasite düğmesi	max_depth	num_leaves	depth
Kategorik değişken	Native destek var; açıkça etkinleştirilir	Native; gradyan istatistiğine göre sıralayıp böler	Sıralı hedef istatistiği (ordered TS) — en olgun çözüm
En güçlü yanı	Olgunluk, kısıt desteği, öngörülebilirlik	Büyük veride eğitim hızı ve düşük bellek	Varsayılanların güvenliği, kategorik zenginlik
En zayıf yanı	Çok büyük veride LightGBM'den yavaş	Küçük veride kolayca ezberler	Eğitim süresi görece uzun olabilir
Tahmin hızı	İyi	İyi	Çok yüksek — simetrik ağaç sayesinde

Önemli çerçeve

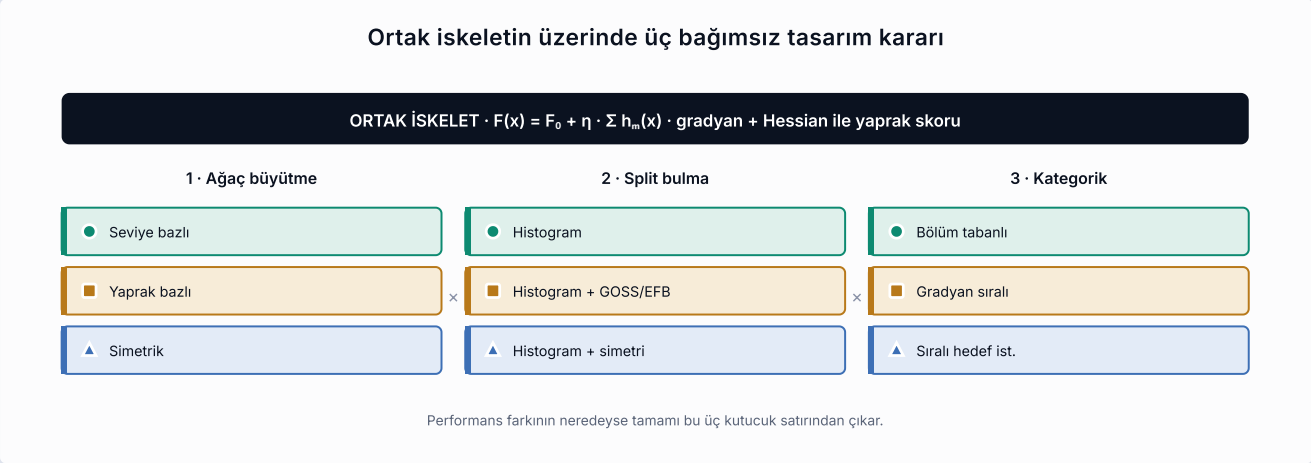
"Hangisi daha iyi?" yanlış sorudur. Üçü de doğru ayarlandığında tabular veride birbirine **çok yakın** sonuç verir. Doğru soru şudur: **bu veri, bu ekip ve bu üretim ortamı** için hangisinin varsayılan davranışı daha az sürprizle sonuçlanır?

02 Ortak temel, ayrışan üç karar

Aynı denklemi paylaşırlar: model, ağaçların ağırlıklı toplamıdır ve her yeni ağaç önceki modelin gradyanına uydurulur.

$$F(x) = F_0 + \eta \cdot [\, h_1(x) + h_2(x) + \dots + h_M(x) \,]$$

Üç kütüphanede de değişmeyen iskelet — fark, h_m'in nasıl kurulduğundadır.



Şekil 2. Performans farklarının neredeyse tamamı bu üç eksenden gelir.

Aynı olanlar

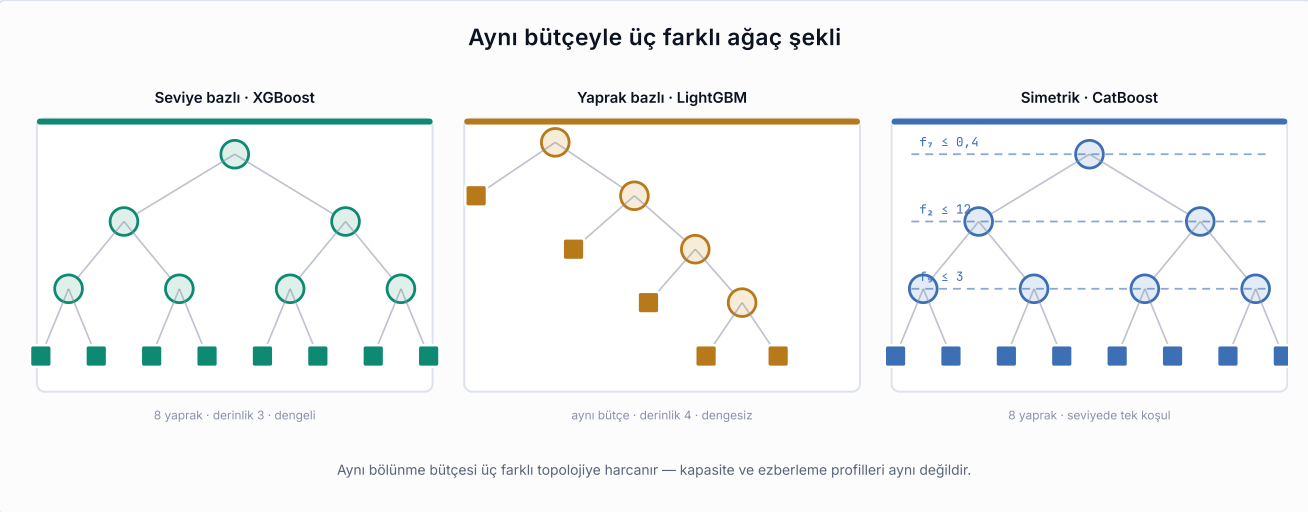
- Kayıp fonksiyonunun gradyanı ve Hessianı üzerinden yaprak skoru hesabı
- Öğrenme oranı (shrinkage) ile her ağacın katkısının küçültülmesi
- Satır ve kolon örneklemesiyle varyans azaltma
- L2 düzenlileştirme ve erken durdurma mantığı
- Eksik değerlerin ayrı bir dal yönü olarak öğrenilmesi
- TreeSHAP ile aynı yöntemle açıklanabilmeleri

Buradan çıkan pratik sonuç

Ortak iskelet aynı olduğu için **tuning sezgin de taşınır**: derinliği azaltmak, düzenlileştirmeyi artırmak, öğrenme oranını düşürüp tur sayısını yükseltmek üç kütüphanede de aynı yönde çalışır. Değişen şey **parametrenin adı ve ölçeğidir**, mantığı değil. 7. ve 8. bölümler bu eşlemeyi verir.

03 Ağaç büyütme stratejisi

Üç kütüphaneyi ayıran **en görünür fark** budur: aynı veriden ürettikleri ağaçların şekli birbirine benzemez.



Şekil 3. Aynı sayıda bölünme, üç farklı topoloji ve üç farklı aşırı öğrenme profili.

Strateji	Seviye bazlı	Yaprak bazlı	Simetrik
Nasıl büyür?	Bir seviyedeki tüm düğümleri açar, sonra bir alt seviyeye iner.	Kazancı en yüksek yaprağı seçip yalnız onu böler.	Bir seviyedeki **tüm** düğümlerde aynı değişken ve eşiği kullanır.
Sonuç	Dengeli, tahmin edilebilir ağaç.	Dengesiz, bazı dalları çok derin ağaç.	Tam dengeli, kusursuz simetrik ağaç.
Avantaj	Aynı bütçede daha az varyans.	Aynı bütçede daha çok kayıp azalması.	Güçlü örtük düzenlileştirme; çok hızlı skorlama.
Risk	Kazançsız dallar da açılır (gamma bunu keser).	Küçük veride derin dallar gürültü ezberler.	Esneklik düşer; ince etkileşimleri kaçırabilir.

En sık yapılan hata: derinlikleri doğrudan eşitlemek

LightGBM'de `num_leaves` ile XGBoost'ta `max_depth` aynı şey değildir. Bir XGBoost ağacı `max_depth=6` ile en fazla **64** yaprak üretir. LightGBM'de `num_leaves=64` vermek çok daha **derin ve dengesiz** bir ağaca izin verir; aynı yaprak sayısı aynı kapasite demek değildir. Pratik kural: `num_leaves` değerini `2^max_depth` değerinin **belirgin biçimde altında** tut.

04 Split adayı bulma ve hızlanma

Bir bölünme aramanın maliyeti, **satır sayısı × değişken sayısı × aday eşik sayısı** ile büyür. Üç kütüphane de bu çarpımın farklı bir terimine saldırır.

Şekil 4. Histogram tüm kütüphanelerde ortaktır; GOSS ve EFB LightGBM'e özgüdür.

Histogram — üçünde de ortak

Sürekli değişken önce kutulara ayrılır; split yalnız kutu kenarlarında aranır. Aday eşik sayısı binlerden birkaç yüze iner. Kutu sayısı ● **XGBoost**'ta `max_bin`, ■ **LightGBM**'de yine `max_bin`, ▲ **CatBoost**'ta `border_count` ile kontrol edilir.

GOSS — büyük gradyanlı satırlara odaklan

Gradyanı küçük olan gözlemler modelin zaten iyi öğrendikleridir. GOSS büyük gradyanlıları tamamen tutar, küçük gradyanlılardan rastgele örnekler ve örneklemeyi telafi etmek için ağırlıklandırır. LightGBM'de `boosting_type="goss"` ile açılır; varsayılan `"gbdt"`'dir.

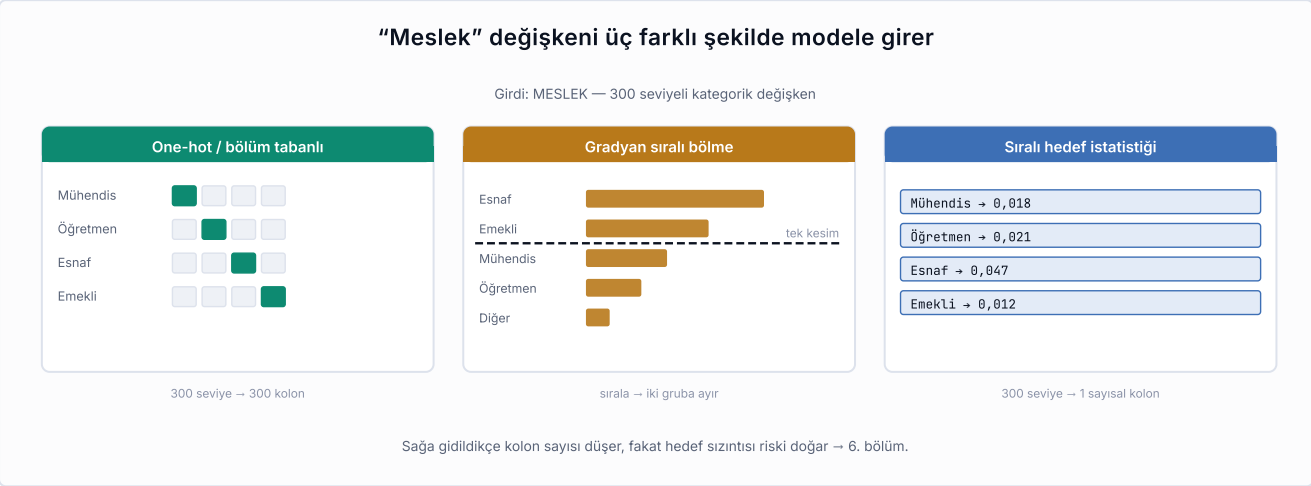
EFB — seyrek değişkenleri paketle

One-hot sonrası oluşan yüzlerce seyrek kolon nadiren aynı satırda birlikte sıfırdan farklıdır. EFB bunları çakışmayacak şekilde tek bir kolonda birleştirir; değişken sayısı terimi doğrudan küçülür. Kredi riskinde **meslek, il, şube, kanal** gibi alanlar one-hot yapıldığında etkisi en çok hissedilen tekniktir.

> **CatBoost'un yaklaşımı farklı**
> CatBoost hızını satır veya kolon eleyerek değil, **ağaç yapısını kısıtlayarak** kazanır. Simetrik ağaçta bir seviyede tek bir bölünme aranır; bu hem arama uzayını küçültür hem de skorlamayı bit maskesi işlemine indirger.

05 Kategorik değişkenler: üç felsefe

Kredi riski verisinin karakterini belirleyen alanların çoğu kategoriktir. Üç kütüphanenin pratikteki en büyük ayrımı burada ortaya çıkar.



Şekil 5. Kardinalite arttıkça one-hot'ın maliyeti patlar; hedef istatistiği sabit kalır.

	XGBoost	LightGBM	CatBoost
Nasıl açılır	`enable_categorical=True` + pandas `category`	`categorical_feature` listesi	`cat_features` listesi
Yöntem	Az kategoride one-hot benzeri; çoksa bölüm (partition) tabanlı	Kategorileri gradyan istatistiğine göre sıralayıp iki gruba ayırır	Sıralı hedef istatistiği (ordered TS) + kategori kombinasyonları
Ayar düğmeleri	`max_cat_to_onehot`, `max_cat_threshold`	`cat_smooth`, `cat_l2`, `min_data_per_group`	`one_hot_max_size`, CTR ayarları
Yüksek kardinalite	İdare eder; ön işleme çoğu zaman daha iyi	İyi, ama aşırı öğrenmeye açık	En güçlü olduğu alan
Otomatik kombinasyon	Yok	Yok	Var — kategori çiftlerini kendiliğinden dener

Kredi riskinde bunun anlamı

Şube kodu (700+), meslek (300+), il–ilçe, ürün ve kanal gibi alanlar one-hot yapıldığında değişken sayısı binlere çıkar; model seyrek kolonlarda gürültü öğrenmeye başlar. Klasik çözüm **WOE kodlamasıdır** — fakat WOE de bir hedef istatistiğidir ve yanlış kurulursa aynı sızıntı problemini taşır. Bir sonraki bölüm tam olarak bunu ele alır.

06 Hedef sızıntısı ve ordered boosting

Hedef kodlaması güçlüdür ama tehlikelidir: bir gözlemin kendi etiketi, o gözleme verilen değerin içine karışırsa model **eğitimde göremeyeceği bir bilgiyi** kullanır.

Şekil 6. Sıralı istatistik, her satır için yalnız kendinden önceki satırları kullanır.

Problem

Bir kategoride yalnızca birkaç gözlem varsa, o kategorinin hedef ortalaması pratikte **satırın kendi etiketinin bir fonksiyonuna** dönüşür. Model bu kolonu çok güçlü bulur; eğitim performansı uçar, OOT'de çöker. Kategori sayısı arttıkça her kategoriye düşen gözlem azalır — yani **kardinalite yükseldikçe sızıntı büyür**.

CatBoost'un çözümü

Veri rastgele bir sıraya konur ve her satırın hedef istatistiği **yalnız kendinden önceki satırlardan** hesaplanır — kendi etiketi asla dâhil edilmez. Aynı fikir ağaç kurma aşamasına da taşınır (**ordered boosting**): gradyanlar, o gözlemi görmemiş bir modelle hesaplanır. Böylece hem hedef sızıntısı hem de tahmin kayması azalır.

> **Kendin kodluyorsan aynı disiplini uygula**
>
> WOE veya target encoding'i elle yapıyorsan istatistiği **yalnız train diliminde** hesapla, fold içinde yeniden üret ve düşük frekanslı kategorileri düzleştir (smoothing). Tüm veri üzerinden bir kez hesaplanan WOE tablosu, kredi riski projelerinde en sık görülen sessiz sızıntı kaynağıdır.

> **Bedeli ne?**
>
> Ordered boosting eğitim süresini uzatır ve küçük veride varyansı bir miktar artırır. CatBoost bu yüzden `boosting_type` için **Ordered** ve **Plain** seçenekleri sunar; büyük veride varsayılan olarak Plain'e yönelir. Sızıntı riski yüksek, veri küçük ve kardinalite yüksekse Ordered'ı bilinçli olarak tercih et.

07 Aşırı öğrenme kontrolü

Aynı niyetin üç kütüphanedeki karşılığı. Soldaki sütun “ne yapmak istiyorum”, sağdakiler “hangi düğmeyi çevireceğim”.

Şekil 7. Tuning sezgisi taşınır; yalnız parametrenin adı ve ölçeği değişir.

Niyet	XGBoost	LightGBM	CatBoost
Ağaç kapasitesini sınırla	max_depth	num_leaves (+ max_depth)	depth
Küçük yaprakları engelle	min_child_weight	min_data_in_leaf, min_sum_hessian_in_leaf	min_data_in_leaf
Kazançsız bölünmeyi engelle	gamma	min_gain_to_split	— (simetri örtük filtre)
Yaprak skorunu yumuşat (L2)	reg_lambda	lambda_l2	l2_leaf_reg
L1 ceza	reg_alpha	lambda_l1	—
Satır örnekle	subsample	bagging_fraction + bagging_freq	subsample (bootstrap_type ile)
Kolon örnekle	colsample_bytree	feature_fraction	rsm
Öğrenme oranı	learning_rate	learning_rate	learning_rate
Tur sayısı	n_estimators	n_estimators / num_boost_round	iterations
Erken durdurma	early_stopping_rounds	early_stopping callback	od_wait, od_type
Sınıf dengesizliği	scale_pos_weight	scale_pos_weight, is_unbalance	scale_pos_weight, auto_class_weights
Monotonluk kısıtı	monotone_constraints	monotone_constraints	monotone_constraints

Ölçekler eşdeğer değildir

`reg_lambda=5` ile `lambda_l2=5` ile `l2_leaf_reg=5` aynı güçte ceza uygulamaz. Aynı sayıyı üç kütüphaneye kopyalayıp “adil karşılaştırma yaptım” demek en yaygın hatadır; her biri için **ayrı arama** gerekir.

08 Başlangıç değerleri

Nadir bad sınıflı, orta/büyük tabular kredi riski verisi için pratik bir başlangıç noktası. Kesin reçete değil, arama alanının merkezi.

Konu	XGBoost	LightGBM	CatBoost
Kapasite	max_depth = 4	num_leaves = 31 · max_depth = 6	depth = 6
Yaprak alt sınırı	min_child_weight = 10	min_data_in_leaf = 200	min_data_in_leaf = 100
L2	reg_lambda = 5	lambda_l2 = 5	l2_leaf_reg = 6
Bölünme eşiği	gamma = 0,1	min_gain_to_split = 0,02	—
Satır örnekleme	subsample = 0,8	bagging_fraction = 0,8 · freq = 1	subsample = 0,8
Kolon örnekleme	colsample_bytree = 0,8	feature_fraction = 0,8	rsm = 0,8
Öğrenme oranı	0,03	0,03	0,03
Tur üst sınırı	n_estimators = 4000	n_estimators = 4000	iterations = 4000
Erken durdurma	100 tur	100 tur	od_wait = 100
Histogram	max_bin = 256	max_bin = 255	border_count = 254
Metrik	eval_metric = "auc"	metric = "auc"	eval_metric = "AUC"

LightGBM için iki kritik ayar

num_leaves tek başına en riskli parametredir; varsayılan 31'den yukarı çıkarken mutlaka `max_depth` ile birlikte sınırla.

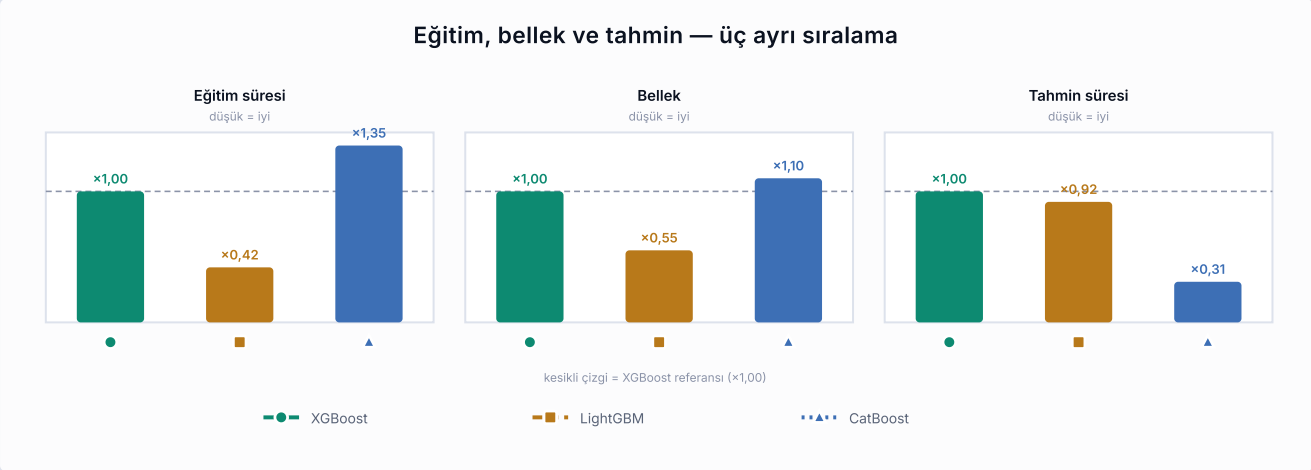
min_data_in_leaf varsayılanı (20) kredi riski için genellikle **çok düşüktür**. Yüz binlerce satırlık portföyde 200–1000 aralığı daha güvenlidir; bu tek değişiklik LightGBM'in ezberleme eğilimini büyük ölçüde bastırır.

CatBoost'ta daha az düğme çevirmek normaldir

CatBoost varsayılanları bilinçli olarak muhafazakârdır ve `learning_rate` değerini veri boyutuna göre kendisi tahmin edebilir. Çoğu projede `depth`, `l2_leaf_reg` ve `iterations` üçlüsünü ayarlamak sonucun büyük kısmını verir.

09 Hız, bellek ve ölçeklenme

Eğitim süresi ile tahmin süresi ayrı konulardır ve sıralamaları farklıdır. Üretimde genellikle ikincisi daha pahalıya mal olur.



Şekil 8. Temsilî değerler; yön göstermek içindir, benchmark sonucu değildir.

Eğitim süresi neden farklılaşır?

■ **LightGBM** hem daha az satır (GOSS) hem daha az kolon (EFB) hem de yaprak bazlı büyümeyle daha az düğüm işler; büyük veride farkı burada açar. ▲ **CatBoost** ordered istatistikleri ve kategori kombinasyonlarını hesaplarken ek iş yapar. ● **XGBoost** ikisinin arasında, öngörülebilir bir noktada durur.

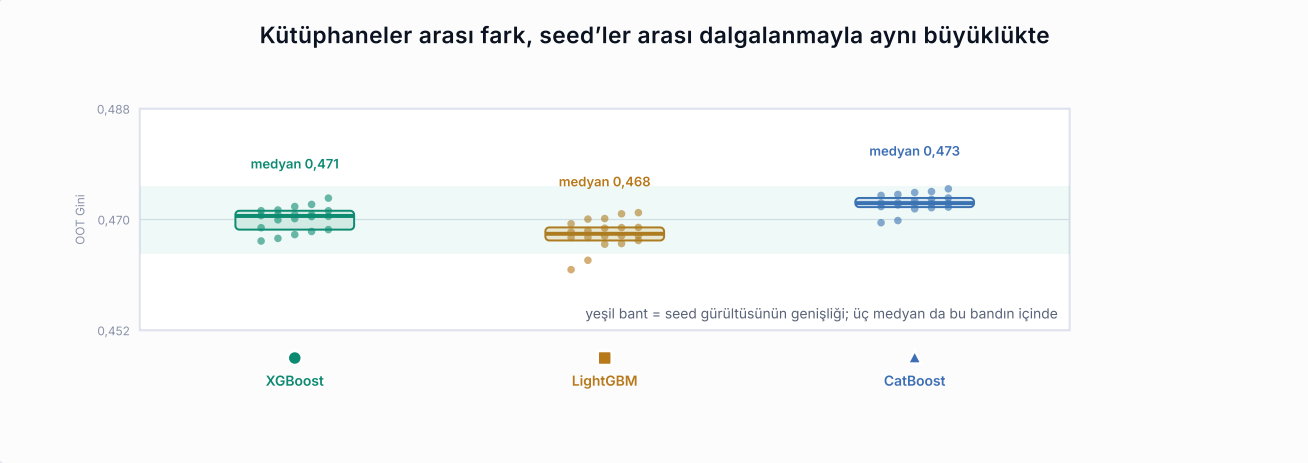
Tahmin süresinde tablo tersine döner

CatBoost'un simetrik ağacında bir gözlemi skorlamak, dallar arasında dolaşmak yerine seviyeler için hesaplanan bit maskesiyle doğrudan yaprak indeksine gitmektir. Gerçek zamanlı karar veren bir başvuru motorunda bu, ölçülebilir bir gecikme avantajıdır.

> **GPU tarafı**
> Üçünün de GPU desteği vardır: ● **XGBoost** `device="cuda"`, ■ **LightGBM** ayrı derleme gerektiren `device="gpu"/"cuda"`, ▲ **CatBoost** `task_type="GPU"`. CatBoost'un GPU uygulaması görece en olgun olanıdır. Ancak birkaç yüz bin satırlık tipik bir kredi riski veri setinde **GPU çoğu zaman kazandırmaz**; veri aktarım maliyeti avantajı yer.

10 Doğruluk farkı gerçekte ne kadar?

Üçü de iyi ayarlandığında tabular veride birbirine çok yakın performans verir. Kritik soru şudur: **gördüğün fark, gürültüden büyük mü?**



Şekil 9. Her nokta bir seed; kutular çeyrekler arası aralığı gösterir (temsilî).

Aynı kütüphaneyi yalnız `random_state` değiştirerek 20 kez eğitmek, OOT Gini'de tipik olarak **±0,004–0,010** bandında bir dalgalanma üretir. İki kütüphane arasındaki fark bu bandın içindeyse, o fark **bir bulgu değildir**.

Karar kuralı

Bir kütüphaneyi diğerine tercih etmeden önce: her birini **en az 5 farklı seed** ve **en az 2 farklı OOT penceresi** ile çalıştır. Farkın medyanı, seed dağılımının genişliğinden büyük değilse seçimini performansa değil; **kategorik işleme, süre, açıklanabilirlik ve ekip aşinalığı** gibi ölçütlere dayandır.

Farkın gerçekten büyüdüğü tek durum

Veri **yüksek kardinaliteli kategorik değişkenlerle doluysa** ve bunlar ham haliyle veriliyorsa, CatBoost ile diğerleri arasındaki fark gürültü bandını aşabilir. Aynı değişkenler dikkatle WOE'ye çevrildiğinde fark yeniden kapanır — yani fark kütüphanenin zekâsından değil, **ön işlemenin kalitesinden** gelir.

11 Eksik değer ve veri hazırlığı

Üçü de eksik değeri kendi başına yönetir; fakat kredi riskinde eksikliğin **kendisi bir sinyaldir** ve bu her zaman otomatik yönetilmemelidir.

Konu	XGBoost	LightGBM	CatBoost
Eksik sayısal	Her bölünmede varsayılan yön öğrenilir	Aynı mantık; `use_missing` ile kapatılabilir	Min/Max uçlarına atanabilir (`nan_mode`)
Eksik kategorik	Ayrı kategori olarak ele alınmalı	Ayrı kategori	Ayrı kategori olarak doğal biçimde işlenir
Ölçekleme gerekir mi?	Hayır	Hayır	Hayır
Aykırı değer	Ağaçlar sıralamaya duyarlı — etkisi sınırlı	Aynı	Aynı
Metin / embedding	Yok	Yok	`text_features`, `embedding_features`

Kredi riskinde eksiklik bir bilgidir

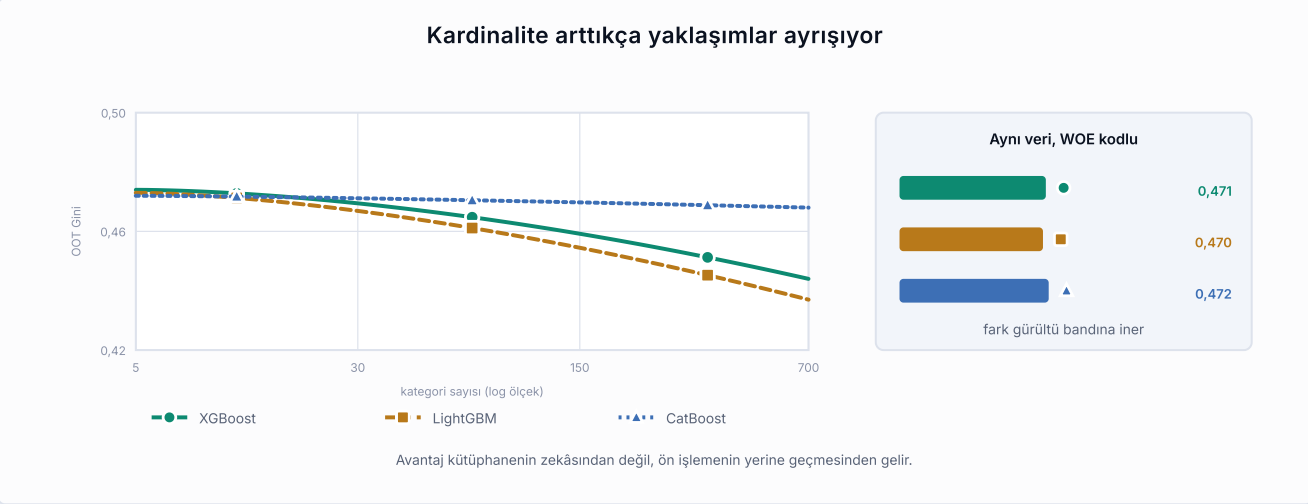
"KKB skoru yok" bir bilinmeyen değil, çoğu zaman **yeni müşteri** demektir; "gelir beyanı yok" bir başvuru kanalını işaret eder. Bu yüzden otomatik yönetimin yanına açık bir `*_IS_MISSING` bayrağı eklemek genellikle hem performansı hem de açıklanabilirliği artırır — ve model dokümanında eksikliğin ne anlama geldiğini yazabilmeni sağlar.

Üretimde eksiklik deseni değişirse

Bir kaynak sistem devre dışı kaldığında eksiklik oranı bir gecede yükselir. Model eksikliği "düşük risk" olarak öğrenmişse skor sessizce kayar. Eksik oranlarını **CSI ile birlikte izle**; bu, üç kütüphanede de aynı derecede geçerli bir üretim riskidir.

12 Kredi riskinde kategorik bolluğu

Tipik bir başvuru veri setinde değişkenlerin üçte biri kategoriktir ve bunların bir kısmı yüzlerce seviyelidir. Kütüphane seçimini en çok etkileyen tek özellik budur.



Şekil 10. Temsilî eğilim: ham kategorik verildiğinde fark açılır, iyi WOE ile kapanır.

Değişken	Tipik seviye	Pratik öneri
Şube kodu	500 – 1000+	CatBoost'a ham ver; diğerlerinde bölge/segment kırılımına indir veya WOE uygula.
Meslek kodu	200 – 400	İş anlamına göre grupla; kalanı "diğer"e topla. Ham hâli her üçünde de riskli.
İl / ilçe	81 / 900+	İl seviyesinde kal; ilçeyi ancak coğrafi risk endeksine dönüştürerek kullan.
Ürün / kampanya	20 – 60	Üçünde de sorunsuz; one-hot bile kabul edilebilir.
Kanal	5 – 10	Düşük kardinalite; fark yaratmaz.

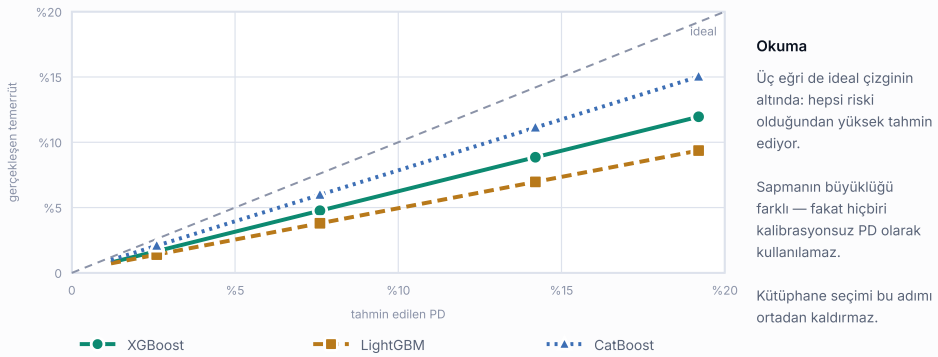
Yönetişim uyarısı: otomatik kolaylık, gözden kaçan risk

CatBoost'un kategorik alanları otomatik işlemesi güçlü bir kolaylıktır; fakat model riski tarafında **"bu değişken skora nasıl dönüştü?"** sorusunun cevabı zorlaşır. Ayrıca otomatik kategori kombinasyonları, ayrımcı olabilecek vekil değişkenleri farkında olmadan üretebilir. Kolaylığı kabul et, fakat üretilen istatistikleri **dokümante et ve gözden geçir**.

13 Kredi riskinde kalibrasyon

Üçü de olasılık üretir; hiçbiri **kalibre edilmiş PD** üretmez. Ama kalibrasyon bozulmasının **şiddeti** ve **yönü** kütüphaneye göre değişir.

Aynı veri, aynı sınıf ağırlığı, üç farklı sapma profili



Okuma

Üç eğri de ideal çizginin altında: hepsi riski olduğundan yüksek tahmin ediyor.

Sapmanın büyüklüğü farklı — fakat hiçbiri kalibrasyonsuz PD olarak kullanılamaz.

Kütüphane seçimi bu adımı ortadan kaldırmaz.

Şekil 11. Temsilî gösterim; gerçek sapma veri ve ayarlara göre değişir.

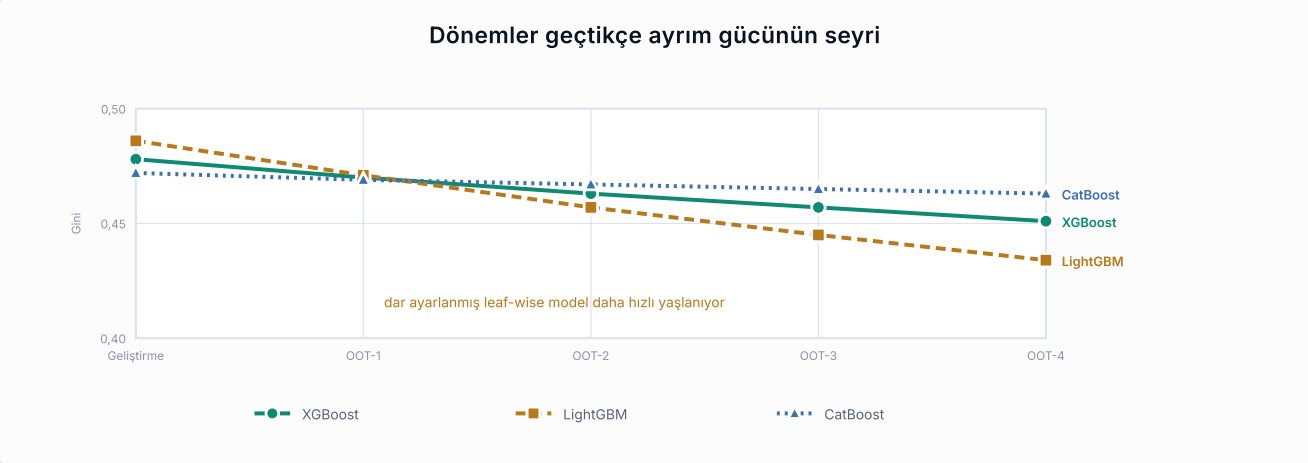
	XGBoost	LightGBM	CatBoost
Tipik eğilim	Orta düzey sapma; L2 ile kontrol edilebilir	Uçlara en çok yığan; derin yapraklar nedeniyle	Genellikle en yumuşak; simetri örtük düzenlileştirme yapar
Kritik parametre	reg_lambda, max_depth	min_data_in_leaf, num_leaves	l2_leaf_reg, depth
Sınıf ağırlığı etkisi	Belirgin	Belirgin	Belirgin

Değişmeyen kural

Hangi kütüphaneyi seçersen seç, `scale_pos_weight` veya sınıf ağırlığı kullandıysan çıktı **PD değildir**. Ayrı bir dilimde kalibrasyon katmanı öğren (lojistik / isotonic), OOT'de doğrula ve dilim bazında gözlenen–beklenen testini yap. Kütüphane seçimi bu adımı ortadan kaldırmaz.

14 Stabilite ve yorumlanabilirlik

Kredi riskinde model bir kez seçilip bitmez; iki yıl boyunca izlenir, savunulur ve denetlenir. Seçim kriterine bu ufku da katmak gerekir.



Şekil 12. Temsilî seyir: dar ayarlanmış leaf-wise model daha hızlı yaşlanma eğilimindedir.

Boyut	XGBoost	LightGBM	CatBoost
Monotonluk kısıtı	Var, olgun	Var, üç yöntem seçeneğiyle	Var
Etkileşim kısıtı	Var	Var (`interaction_constraints`)	Sınırlı
SHAP desteği	TreeSHAP	TreeSHAP	Yerleşik SHAP çıktısı
Skor stabilitesi	Yüksek	Ayara duyarlı	Yüksek (simetri sayesinde)
Ekip aşinalığı	En yaygın	Yaygın	Görece yeni

Denetim tarafının sorusu

Bağımsız validasyon biriminin ilk soracağı şey kütüphane adı değil, **“aynı sonucu yeniden üretebiliyor musunuz?”** olacaktır. Üçünde de seed’i, sürümü, veri kesitini ve parametreleri kilitle. LightGBM ve XGBoost’ta iş parçacığı sayısı bile sonucu az miktarda değiştirebilir; kritik modelde bunu da sabitle ve test et.

15 Python: aynı problem, üç kütüphane

Aynı veri, aynı bölme, aynı metrik. Yalnız parametre adları değişiyor.

```python
# ---------- ORTAK KURULUM ----------
neg, pos = (y_tr == 0).sum(), (y_tr == 1).sum()
spw = neg / max(pos, 1)
cat_cols = ["SUBE", "MESLEK", "IL", "URUN", "KANAL"]

# ---------- XGBoost ----------
from xgboost import XGBClassifier
X_tr_x = X_tr.copy(); X_va_x = X_va.copy()
for c in cat_cols:                        # native kategorik için category dtype şart
    X_tr_x[c] = X_tr_x[c].astype("category")
    X_va_x[c] = X_va_x[c].astype(X_tr_x[c].dtype)

xgb = XGBClassifier(
    objective="binary:logistic", eval_metric="auc", tree_method="hist",
    enable_categorical=True, max_cat_to_onehot=8,
    n_estimators=4000, learning_rate=0.03, max_depth=4,
    min_child_weight=10, gamma=0.1, reg_lambda=5.0,
    subsample=0.8, colsample_bytree=0.8, scale_pos_weight=spw,
    early_stopping_rounds=100, random_state=42, n_jobs=8
)
xgb.fit(X_tr_x, y_tr, eval_set=[(X_va_x, y_va)], verbose=False)

# ---------- LightGBM ----------
import lightgbm as lgb
lgbm = lgb.LGBMClassifier(
    objective="binary", metric="auc",
    n_estimators=4000, learning_rate=0.03,
    num_leaves=31, max_depth=6,        # ikisini BİRLİKTE sınırla
    min_data_in_leaf=200,               # varsayılan 20 kredi riski için düşük
    lambda_l2=5.0, min_gain_to_split=0.02,
    bagging_fraction=0.8, bagging_freq=1, feature_fraction=0.8,
    scale_pos_weight=spw, random_state=42, n_jobs=8
)
lgbm.fit(X_tr, y_tr, eval_set=[(X_va, y_va)],
         categorical_feature=cat_cols,
         callbacks=[lgb.early_stopping(100, verbose=False)])

# ---------- CatBoost ----------
from catboost import CatBoostClassifier, Pool
pool_tr = Pool(X_tr, y_tr, cat_features=cat_cols)
pool_va = Pool(X_va, y_va, cat_features=cat_cols)

cb = CatBoostClassifier(
    loss_function="Logloss", eval_metric="AUC",
    iterations=4000, learning_rate=0.03, depth=6,
    l2_leaf_reg=6.0, min_data_in_leaf=100, rsm=0.8,
    scale_pos_weight=spw, od_type="Iter", od_wait=100,
    random_seed=42, verbose=False
)
cb.fit(pool_tr, eval_set=pool_va, use_best_model=True)
```

Adil karşılaştırma iskeleti

Tek bir seed ile alınan tek bir Gini, karşılaştırma değildir. Aşağıdaki iskelet farkı **gürültüyle birlikte** raporlar.

```
import numpy as np, pandas as pd
from sklearn.metrics import roc_auc_score, average_precision_score, brier_score_loss

def degerlendir(model, X, y, kind):
    p = model.predict_proba(X)[:, 1]
    auc = roc_auc_score(y, p)
    return {
        "kutuphane": kind,
        "gini": 2 * auc - 1,
        "pr_auc": average_precision_score(y, p),
        "brier": brier_score_loss(y, p),      # kalibrasyonu da ölç
        "top1_capture": y[p ≥ np.quantile(p, 0.99)].mean() / y.mean()
    }

sonuc = []
for seed in [42, 7, 2024, 101, 555]:          # seed varyansını ölç
    for oot in oot_pencereleri:                # en az iki farklı dönem
        for ad, kur in kurucular.items():      # {"xgb": fn, "lgbm": fn, "cat": fn}
            m = kur(seed)                      # her biri KENDİ aramasıyla ayarlanmış
            m.fit(*egitim_verisi)
            r = degerlendir(m, *oot, ad)
            r["seed"], r["oot"] = seed, oot.ad
            sonuc.append(r)

df = pd.DataFrame(sonuc)
ozet = df.groupby("kutuphane")["gini"].agg(["median", "std", "min", "max"])
print(ozet)
# Karar kuralı: medyan farkı, seed std'sinin ~2 katından küçükse "fark yok" say.
```

İskeletteki üç bilinçli tercih

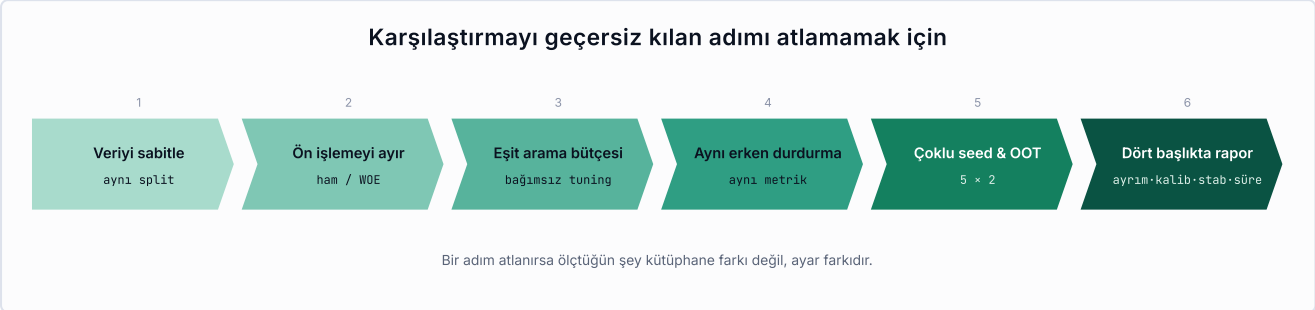
Brier metriği listede: kalibrasyonu ölçmeden yapılan karşılaştırma yalnız sıralamayı görür ve LightGBM'in uçlara yığma eğilimini gizler.

top1_capture listede: operasyonel kapasitenin çalıştığı dilimdeki farkı toplam Gini gizleyebilir.

Her kütüphane kendi aramasıyla ayarlanıyor: aynı parametreleri üçüne kopyalamak, en iyi ayarı bulunan değil, **varsayılanı en şanslı olan** kütüphaneyi kazandırır.

16 Adil karşılaştırma protokolü

Kütüphane karşılaştırmasının çoğu, karşılaştırma değil **ayar farkı ölçümüdür**. Aşağıdaki sıra bunu engeller.



Şekil 13. Her adım, bir önceki adımın çıktısını sabitleyerek ilerler.

1. **Veriyi sabitle.** Aynı train / validation / OOT bölmesi, aynı müşteri bazlı split, aynı bad tanımı. Bölme farkı, kütüphane farkından büyüktür.
2. **Ön işlemeyi ayır.** Kategorikleri iki kolda dene: (a) ham, native destekle; (b) WOE ile kodlanmış. Bu iki kolu karıştırma.
3. **Her kütüphaneye eşit arama bütçesi ver.** Aynı sayıda deneme, aynı arama yöntemi. Kopyalanmış parametre değil, **bağımsız arama**.
4. **Erken durdurmayı aynı kur.** Aynı validation seti, aynı metrik, aynı sabır.
5. **Çoklu seed ve çoklu OOT ile tekrarla.** En az 5 seed, en az 2 dönem.
6. **Dört başlıkta raporla:** ayrım, kalibrasyon, stabilite, süre. Tek metrikli karşılaştırma karar için yetersizdir.

> **Rapor şablonu**
> Her kütüphane için tek satır: `medyan Gini ± std` · `PR-AUC` · `Brier` · `üst %1 yakalama` · `PSI` · `eğitim süresi` · `skorlama gecikmesi`. Kararı bu tabloya bakarak ver; tek bir Gini sayısına bakarak değil.

17 Hangisini ne zaman seçmeli?

Performanslar yakınsa karar, verinin ve ortamın özelliklerine göre verilir.

Şekil 14. Akış bir kural değil, başlangıç noktasıdır; nihai karar ölçümle verilir.

Durum	İlk deneme	Neden
Çok sayıda yüksek kardinaliteli kategorik, sınırlı ön işleme zamanı	▲ CatBoost	Ordered TS sızıntısız çalışır; varsayılanlar güvenli.
Milyonlarca satır, sık yeniden eğitim, süre kritik	■ LightGBM	Eğitim süresi ve bellekte belirgin avantaj.
Model riski onayı ağır, kısıt ve dokümantasyon önemli	● XGBoost	En olgun ekosistem; kısıt ve araç desteği en geniş.
Gerçek zamanlı başvuru motoru, gecikme bütçesi dar	▲ CatBoost	Simetrik ağaç skorlamayı çok hızlandırır.
Küçük veri (< 50 bin satır)	● XGBoost / ▲ CatBoost	Leaf-wise büyüme küçük veride kolayca ezberler.
Değişkenler zaten WOE'ye çevrilmiş skorkart hattı	Fark küçülür	Kategorik avantajı ortadan kalkar; süre ve ekip aşinalığı belirler.

Pratik öneri

Üçünü de kur, ama **birini şampiyon seç**. Ekip tek bir kütüphanede derinleşince elde edilen kazanç, kütüphaneler arasındaki farktan büyüktür. Diğer ikisini **meydan okuyan** olarak yılda bir kez aynı protokolle yeniden çalıştır.

18 Yaygın tuzaklar

Karşılaştırmayı ya da modeli sessizce bozan hatalar.

Tuzak	Ne olur?	Doğrusu
num_leaves = 2^max_depth	LightGBM aşırı derin ağaçlar kurar; OOT çöker.	num_leaves'i bu sınırın belirgin altında tut, max_depth ile birlikte sınırla.
min_data_in_leaf varsayılanda bırakmak	Nadir bad sınıfında 20 gözlemlik yapraklar oluşur.	Büyük portföyde 200–1000 aralığını dene.
Aynı parametreleri üçüne kopyalamak	Ölçekler farklı olduğu için ayar değil, şans ölçülür.	Her kütüphaneye eşit bütçeyle bağımsız arama.
WOE'yi tüm veride hesaplamak	Sessiz hedef sızıntısı; validasyon iyimser çıkar.	Yalnız train diliminde, fold içinde yeniden üret.
Kategorikleri sayısal kodla vermek	Model 5 > 3 gibi anlamsız bir sıralama öğrenir.	Native kategorik desteğini kullan veya WOE/one-hot uygula.
Tek seed ile karar vermek	Gürültü, bulgu sanılır.	Çoklu seed ve çoklu OOT ile medyan ± std raporla.
Sadece AUC raporlamak	Kalibrasyon ve üst dilim farkı görünmez.	Gini, PR-AUC, Brier, üst dilim yakalama birlikte.
OOT'yi erken durdurmada kullanmak	Üç kütüphanede de aynı hata; performans iyimser.	Ayrı validation; OOT yalnız bir kez ölçülür.
GPU'yu otomatik kazanç sanmak	Küçük veride aktarım maliyeti avantajı siler.	Aynı ayarla CPU–GPU karşılaştırması yap.
Sürümü kilitlememek	Varsayılan değişikliği modeli sessizce oynatır.	Sürümü ve seed'i model dokümanına yaz, ortamı sabitle.

Hepsinin ortak paydası

Listedeki hataların hiçbiri kütüphane hatası değildir. Üçü de kendilerinden isteneni yapar; sorun **ne istediğimizi yanlış ifade etmemizden** çıkar.

Hızlı referans kartı

Tek sayfada üç kütüphane.

	XGBoost	LightGBM	CatBoost
Ağaç şekli	Dengeli (depthwise)	Dengesiz (leaf-wise)	Simetrik (oblivious)
Kapasite düğmesi	max_depth = 3–6	num_leaves = 15–63	depth = 4–8
Yaprak alt sınırı	min_child_weight	min_data_in_leaf	min_data_in_leaf
L2	reg_lambda	lambda_l2	l2_leaf_reg
Kolon örnekleme	colsample_bytree	feature_fraction	rsm
Kategorik	enable_categorical	categorical_feature	cat_features
Erken durdurma	early_stopping_rounds	early_stopping callback	od_wait
GPU	device = "cuda"	device = "gpu"	task_type = "GPU"
Eğitim hızı	Orta	En hızlı	Orta / yavaş
Tahmin hızı	Orta	Orta	En hızlı
Ezberleme riski	Orta	Yüksek	Düşük
Varsayılan güvenliği	Orta	Düşük	Yüksek
İlk tercih olduğu yer	Kısıt ve onay ağırlıklı hatlar	Büyük veri, sık eğitim	Kategorik zengin veri, düşük gecikme

Tek cümlelik özet

XGBoost öngörülebilirliği, **LightGBM** hızı, **CatBoost** kategorik veriyi ve güvenli varsayılanları satın aldığın yerdir. Kredi riski hattında üçü de kabul edilebilir bir modeldir; farkı yaratan kütüphane değil, **validasyon tasarımı, kalibrasyon ve izleme disiplinidir**.

19 Kaynaklar

Resmî dokümantasyon ve yöntemleri tanıtan özgün makaleler.

1. XGBoost Documentation — Parameters, Tree Methods, Categorical Data
   `xgboost.readthedocs.io/en/stable/`
2. LightGBM Documentation — Parameters, Features, Parameters Tuning
   `lightgbm.readthedocs.io/en/stable/`
3. CatBoost Documentation — Parameters, Categorical Features, Ordered Boosting
   `catboost.ai/docs/`
4. Chen & Guestrin — *XGBoost: A Scalable Tree Boosting System* (KDD 2016)
   `arxiv.org/abs/1603.02754`
5. Ke et al. — *LightGBM: A Highly Efficient Gradient Boosting Decision Tree* (NeurIPS 2017)
   `GOSS ve EFB tekniklerinin tanıtıldığı makale`
6. Prokhorenkova et al. — *CatBoost: unbiased boosting with categorical features* (NeurIPS 2018)
   `arxiv.org/abs/1706.09516`
7. BCDataWorks Çalışma Notları № 02 — *XGBoost Hiperparametreleri*
   `Parametre bazlı derinlemesine anlatım ve kredi riski bölümleri`

Kapsam notu

Bu doküman **tabular ikili sınıflandırma** ve kredi riski bağlamına odaklanır. Sıralama (ranking), çok sınıflı, sağkalım, zaman serisi ve dağıtık eğitim senaryolarında üç kütüphanenin göreli konumu değişebilir.

Son söz

Kütüphane seçimi, bir kredi riski modelinin başarısını belirleyen ilk on faktörden biri bile değildir. Hedef tanımı, validasyon tasarımı, değişken kalitesi, kalibrasyon ve izleme; üçü arasındaki farkın kat kat üzerinde etki taşır. Bu dokümanın amacı seçimi **hızlıca ve gerekçeli biçimde kapatıp** asıl işe dönmektir.

BCDataWorks Çalışma Notları № 03 · Ağustos 2026

— ÇALIŞMA SONU —